Thread migration in the presence of pointers

Cronk, D. Haines, M. Mehrotra, P.
Dept. of Comput. Sci., Coll. of William & Mary, Williamsburg, VA, USA;

This paper appears in: **System Sciences, 1997, Proceedings of the Thirtieth Hawaii International Conference on**

Meeting Date: 01/07/1997 - 01/10/1997
Publication Date: 7-10 Jan. 1997
Location: Wailea, HI USA
On page(s): 292 - 298 vol.1
Volume: 1
Reference Cited: 14
Number of Pages: 6 vol. (xvii+766+xv+716+620+xiv+546+xiii+720+ix+275)
Inspec Accession Number: 5907178

Abstract:
Dynamic migration of lightweight **threads** supports both data locality and load balancing. However, migrating **threads** that contain pointers referencing data in both the **stack** and **heap** remains an open problem. We describe a technique by which **threads** with pointers referencing both **stack** and non shared **heap** data can be migrated such that the pointers remain valid after migration. As a result, **threads** containing pointers can now be migrated between processors in a homogeneous distributed memory environment

Index Terms:
data structures distributed memory systems parallel programming program control structures resource allocation data locality dynamic migration homogeneous distributed memory environment lightweight threads load balancing migrating threads non shared heap data pointer referencing stack data thread migration

The Augmint multiprocessor simulation toolkit for Intel x86 architectures

Nguyen, A.-T. Michael, M. Sharma, A. Torrellas, J.
Center for Supercomput. Res. & Dev., Illinois Univ., Urbana, IL, USA;

This paper appears in: **Computer Design: VLSI in Computers and Processors, 1996. ICCD '96. Proceedings., 1996 IEEE International Conference on**

Meeting Date: 10/07/1996 - 10/09/1996
Publication Date: 7-9 Oct. 1996
Location: Austin, TX USA
On page(s): 486 - 490
Reference Cited: 13
Number of Pages: xx+589
Inspec Accession Number: 5437709

Abstract:
Most publicly available simulation tools only simulate RISC architectures. These tools cannot capture the instruction mix and memory reference patterns of CISC architectures. We present an overview of Augmint, an execution driven multiprocessor simulation toolkit that fills this gap by supporting Intel x86 architectures. Augmint also supports trace driven simulation for uniprocessors as well as multiprocessors, with minor effort on the part of simulator developers. Augmint runs m4 macro extended C and C++ applications such as those in the SPLASH and SPLASH-2 benchmark suites. Augmint supports a thread based programming model with shared global address space and private stack space. Augmint supports a simulator interface compatible with that of the MINT simulation toolkit for MIPS architectures, thus allowing the reuse of most architecture simulators written for MINT. Augmint simulations run on x8d based uniprocessor systems under Unix or Windows NT. The source code of Augmint is publicly available from http://www.csrd.uiuc.edu/iacoma/augmint

Index Terms:
instruction sets multiprocessing systems parallel architectures parallel programming virtual machines Augmint multiprocessor simulation toolkit C++ applications CISC architectures Intel x86 architectures MINT simulation toolkit SPLASH-2 benchmark suites architecture simulators execution driven multiprocessor simulation toolkit instruction mix m4 macro extended C memory reference patterns private stack space publicly available simulation tools shared global address

Your search matched **2** of **1114111** documents.
A maximum of **500** results are displayed, **15** to a page, sorted by **Relevance** in **Descending** order.

Refine This Search:
You may refine your search by editing the current search expression or entering a new one in the text box.

thread <and> stack <and> local Search

☐ Check to search within this result set

Results Key:
JNL = Journal or Magazine **CNF** = Conference **STD** = Standard

---

1 **HRTEM study of strain energy distribution on the self-organized In-rich quantum dots of the InGaN/GaN quantum wells**
Yen-Sheng Lin; Kung-Jen Ma; Chih-Chung Yang; Weirich, T.E.;
Numerical Simulation of Optoelectronic Devices, 2004. NUSOD '04. Proceedings of the 4th International Conference on , 24-26 Aug. 2004
Pages:88 - 89

[Abstract] [PDF Full-Text (290 KB)] **IEEE CNF**

---

2 **Building distributed embedded systems with RTLinux-GPL**
Perez, S.; Vila, J.;
Emerging Technologies and Factory Automation, 2003. Proceedings. ETFA '03. IEEE Conference , Volume: 1 , 16-19 Sept. 2003
Pages:161 - 168 vol.1

[Abstract] [PDF Full-Text (619 KB)] **IEEE CNF**

Terms used **local** and **global** and **thread stack** and **heap** Found **41** of **148,786**

Sort results by: relevance

Display results: expanded form

Save results to a Binder

Search Tips

Open results in a new window

Try an Advanced Search

Try this search in The ACM Guide

Results 1 - 20 of 41 Result page: **1** 2 3 next

Relevance scale

1 Thread-local heaps for Java

Tamar Domani, Gal Goldshtein, Elliot K. Kolodner, Ethan Lewis, Erez Petrank, Dafna Sheinwald

June 2002 **ACM SIGPLAN Notices , Proceedings of the 3rd international symposium on Memory management**, Volume 38 Issue 2 supplement

Full text available: pdf(220.38 KB) Additional Information: full citation, abstract, references, citings, index terms

We present a memory management scheme for Java based on thread-local heaps. Assuming most objects are created and used by a single thread, it is desirable to free the memory manager from redundant synchronization for thread-local objects. Therefore, in our scheme each thread receives a partition of the heap in which it allocates its objects and in which it does local garbage collection without synchronization with other threads. We dynamically monitor to determine which objects are local and whi ...

Keywords: JVM, Java, garbage collection, local garbage collection, locality, scalable garbage collection

2 Reliability: Application-level checkpointing for shared memory programs

Greg Bronevetsky, Daniel Marques, Keshav Pingali, Peter Szwed, Martin Schulz

October 2004 **Proceedings of the 11th international conference on Architectural support for programming languages and operating systems**

Full text available: pdf(235.77 KB) Additional Information: full citation, abstract, references, index terms

Trends in high-performance computing are making it necessary for long-running applications to tolerate hardware faults. The most commonly used approach is checkpoint and restart (CPR) - the state of the computation is saved periodically on disk, and when a failure occurs, the computation is restarted from the last saved state. At present, it is the responsibility of the programmer to instrument applications for CPR.Our group is investigating the use of compiler technology to instrument codes to ...

Keywords: checkpointing, fault-tolerance, openMP, shared-memory programs

3 Implementing an on-the-fly garbage collector for Java

Tamar Domani, Elliot K. Kolodner, Ethan Lewis, Eliot E. Salant, Katherine Barabash, Itai Lahan, Yossi Levanoni, Erez Petrank, Igor Yanorer

October 2000 **ACM SIGPLAN Notices , Proceedings of the 2nd international symposium**

on Memory management, Volume 36 Issue 1
Full text available: pdf(1.33 MB) Additional Information: full citation, abstract, citings, index terms

Java uses garbage collection (GC) for the automatic reclamation of computer memory no longer required by a running application. GC implementations for Java Virtual Machines (JVM) are typically designed for single processor machines, and do not necessarily perform well for a server program with many threads running on a multiprocessor. We designed and implemented an on-the-fly GC, based on the algorithm of Doligez, Leroy and Gonthier [13, 12] (DLG), for Java in this environment. An *on-the-f ...*

***Keywords**: Java, concurrent garbage collection, garbage collection, memory management, on-the-fly garbage collection, programming languages*

4 Atomizer: a dynamic atomicity checker for multithreaded programs
Cormac Flanagan, Stephen N Freund
January 2004 **ACM SIGPLAN Notices , Proceedings of the 31st ACM SIGPLAN-SIGACT symposium on Principles of programming languages**, Volume 39 Issue 1
Full text available: pdf(195.13 KB) Additional Information: full citation, abstract, references, citings, index terms

Ensuring the correctness of multithreaded programs is difficult, due to the potential for unexpected interactions between concurrent threads. Much previous work has focused on detecting race conditions, but the absence of race conditions does not by itself prevent undesired thread interactions. We focus on the more fundamental non-interference property of *atomicity*; a method is atomic if its execution is not affected by and does not interfere with concurrently-executing threads. Atomic me ...

Keywords: atomicity, dynamic analysis, reduction

5 Cycles to recycle: garbage collection to the IA-64
Richard L. Hudson, J. Elliot Moss, Sreenivas Subramoney, Weldon Washburn
October 2000 **ACM SIGPLAN Notices , Proceedings of the 2nd international symposium on Memory management**, Volume 36 Issue 1
Full text available: pdf(1.25 MB) Additional Information: full citation, abstract, citings, index terms

The IA-64, Intel's 64-bit instruction set architecture, exhibits a number of interesting architectural features. Here we consider those features as they relate to supporting garbage collection (GC). We aim to assist GC and compiler implementors by describing how one may exploit features of the IA-64. Along the way, we record some previously unpublished object scanning techniques, and offer novel ones for object allocation (suggesting some simple operating system support that would simplify it ...

6 A memory-efficient real-time non-copying garbage collector
Tian F. Lim, Przemysław Pardyak, Brian N. Bershad
October 1998 **ACM SIGPLAN Notices , Proceedings of the 1st international symposium on Memory management**, Volume 34 Issue 3
Full text available: pdf(1.56 MB) Additional Information: full citation, abstract, references, citings, index terms

Garbage collectors used in embedded systems such as Personal Java and Inferno or in operating systems such as SPIN must operate with limited resources and minimize their impact on application performance. Consequently, they must maintain short real-time pauses, low overhead, and a small memory footprint. Most garbage collectors, including the Treadmill algorithm, are inadequate because they sacrifice space for time. We have implemented a new Treadmill variant that provides good memory utilizatio ...

Keywords: garbage collection, operating systems, real-time, treadmill

7 Efficient object sampling via weak references

Ole Agesen, Alex Garthwaite

October 2000 **ACM SIGPLAN Notices , Proceedings of the 2nd international symposium on Memory management**, Volume 36 Issue 1

Full text available: pdf(743.52 KB) Additional Information: full citation, abstract, citings, index terms

The performance of automatic memory management may be improved if the policies used in allocating and collecting objects had knowledge of the lifetimes of objects. To date, approaches to the pretenuring of objects in older generations have relied on profile-driven feedback gathered from trace runs. This feedback has been used to specialize allocation sites in a program. These approaches suffer from a number of limitations. We propose an alternative that through efficient sampling of objects a ...

8 Concurrency: Write barrier elision for concurrent garbage collectors

Martin T. Vechev, David F. Bacon

October 2004 **Proceedings of the 4th international symposium on Memory management**

Full text available: pdf(490.73 KB) Additional Information: full citation, abstract, references, index terms

Concurrent garbage collectors require write barriers to preserve consistency, but these barriers impose significant direct and indirect costs. While there has been a lot of work on optimizing write barriers, we present the first study of their elision in a concurrent collector. We show conditions under which write barriers are redundant, and describe how these conditions can be applied to both incremental update or snapshot-at-the-beginning barriers. We then evaluate the potential for write b ...

Keywords: concurrent garbage collection, write barrier

9 Supporting dynamic data structures on distributed-memory machines

Anne Rogers, Martin C. Carlisle, John H. Reppy, Laurie J. Hendren

March 1995 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 17 Issue 2

Full text available: pdf(2.05 MB) Additional Information: full citation, abstract, references, citings, index terms, review

Compiling for distributed-memory machines has been a very active research area in recent years. Much of this work has concentrated on programs that use arrays as their primary data structures. To date, little work has been done to address the problem of supporting programs that use pointer-based dynamic data structures. The techniques developed for supporting SPMD execution of array-based programs rely on the fact that arrays are statically defined and directly addressable. Recursive data s ...

Keywords: dynamic data structures

10 A generational mostly-concurrent garbage collector

Tony Printezis, David Detlefs

October 2000 **ACM SIGPLAN Notices , Proceedings of the 2nd international symposium on Memory management**, Volume 36 Issue 1

Full text available: pdf(1.67 MB) Additional Information: full citation, abstract, citings, index terms

This paper reports our experiences with a mostly-concurrent incremental garbage collector, implemented in the context of a high performance virtual machine for the Java™ programming language. The garbage collector is based on the "mostly parallel" collection

algorithm of Boehm *et al.* and can be used as the old generation of a generational memory system. It overloads efficient write-barrier code already generated to support generational garbage collection to also ident ...

11 Session I: Extending the Haskell foreign function interface with concurrency
Simon Marlow, Simon Peyton Jones, Wolfgang Thaller
September 2004 **Proceedings of the ACM SIGPLAN workshop on Haskell**

Full text available: pdf(192.06 KB) Additional Information: full citation, abstract, references, index terms

A Haskell system that includes both the Foreign Function Interface and the Concurrent Haskell extension must consider how Concurrent Haskell threads map to external Operating System threads for the purposes of specifying in which thread a foreign call is made.Many concurrent languages take the easy route and specify a one-to-one correspondence between the language's own threads and external OS threads. However, OS threads tend to be expensive, so this choice can limit the performance and scalabi ...

12 Operating systems security: On the effectiveness of address-space randomization
Hovav Shacham, Matthew Page, Ben Pfaff, Eu-Jin Goh, Nagendra Modadugu, Dan Boneh
October 2004 **Proceedings of the 11th ACM conference on Computer and communications security**

Full text available: pdf(193.68 KB) Additional Information: full citation, abstract, references, index terms

Address-space randomization is a technique used to fortify systems against buffer overflow attacks. The idea is to introduce artificial diversity by randomizing the memory location of certain system components. This mechanism is available for both Linux (via PaX ASLR) and OpenBSD. We study the effectiveness of address-space randomization and find that its utility on 32-bit architectures is limited by the number of bits available for address randomization. In particular, we demonstrate a < ...

Keywords: address-space randomization, automated attacks, diversity

13 Escape analysis for Java
Jong-Deok Choi, Manish Gupta, Mauricio Serrano, Vugranam C. Sreedhar, Sam Midkiff
October 1999 **ACM SIGPLAN Notices , Proceedings of the 14th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications**, Volume 34 Issue 10

Full text available: pdf(1.85 MB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents a simple and efficient data flow algorithm for escape analysis of objects in Java programs to determine (i) if an object can be allocated on the stack; (ii) if an object is accessed only by a single thread during its lifetime, so that synchronization operations on that object can be removed. We introduce a new program abstraction for escape analysis, the connection graph, that is used to establish reachability relationships between objects and object ref ...

14 An on-the-fly reference counting garbage collector for Java
Yossi Levanoni, Erez Petrank
October 2001 **ACM SIGPLAN Notices , Proceedings of the 16th ACM SIGPLAN conference on Object oriented programming, systems, languages, and applications**, Volume 36 Issue 11

Full text available: pdf(280.30 KB) Additional Information: full citation, abstract, references, citings, index terms

Reference counting is not naturally suitable for running on multiprocessors. The update of pointers and reference counts requires atomic and synchronized operations. We present a

novel reference counting algorithm suitable for a multiprocessor that does not require any synchronized operation in its write barrier (not even a compare-and-swap type of synchronization). The algorithm is efficient and may complete with any tracing algorithm.

15 Sharing and protection in a single-address-space operating system
Jeffrey S. Chase, Henry M. Levy, Michael J. Feeley, Edward D. Lazowska
November 1994 **ACM Transactions on Computer Systems (TOCS)**, Volume 12 Issue 4

Full text available: pdf(2.87 MB) Additional Information: full citation, abstract, references, citings, index terms

This article explores memory sharing and protection support in Opal, a single-address-space operating system designed for wide-address (64-bit) architectures. Opal threads execute within protection domains in a single shared virtual address space. Sharing is simplified, because addresses are context independent. There is no loss of protection, because addressability and access are independent; the right to access a segment is determined by the protection domain in which a thread executes. T ...

Keywords: 64-bit architectures, capability-based systems, microkernel operating systems, object-oriented database systems, persistent storage, protection, single-address-space operating systems, wide-address architectures

16 A foundation for an efficient multi-threaded scheme system
Suresh Jagannathan, Jim Philbin
January 1992 **ACM SIGPLAN Lisp Pointers , Proceedings of the 1992 ACM conference on LISP and functional programming**, Volume V Issue 1

Full text available: pdf(1.19 MB) Additional Information: full citation, abstract, references, citings, index terms

We have built a parallel dialect of Scheme called STING that differs from its contemporaries in a number of important respects. STING is intended to be used as an operating system substrate for modern parallel programming languages. The basic concurrency management objects is STING are first-class lightweight threads of control and virtual processors (VPs). Unlike high-level concurrency structures, STING threads and VPs are not encumbered by complex synchronization protocols. ...

17 Whole-program optimization for time and space efficient threads
Dirk Grunwald, Richard Neves
September 1996 **Proceedings of the seventh international conference on Architectural support for programming languages and operating systems**, Volume 31 , 30 Issue 9 , 5

Full text available: pdf(1.11 MB) Additional Information: full citation, abstract, references, citings, index terms

Modern languages and operating systems often encourage programmers to use *threads,* or independent control streams, to mask the overhead of some operations and simplify program structure. Multitasking operating systems use threads to mask communication latency, either with hardwares devices or users. Client-server applications typically use threads to simplify the complex control-flow that arises when multiple clients are used. Recently, the scientific computing community has started using ...

18 Improving server software support for simultaneous multithreaded processors
Luke K. McDowell, Susan J. Eggers, Steven D. Gribble
June 2003 **ACM SIGPLAN Notices , Proceedings of the ninth ACM SIGPLAN symposium on Principles and practice of parallel programming**, Volume 38 Issue 10

Full text available: pdf(218.63 KB) Additional Information: full citation, abstract, references, index terms

Simultaneous multithreading (SMT) represents a fundamental shift in processor capability. SMT's ability to execute multiple threads simultaneously within a single CPU offers tremendous potential performance benefits. However, the structure and behavior of software affects the extent to which this potential can be achieved. Consequently, just like the earlier arrival of multiprocessors, the advent of SMT processors prompts a needed re-evaluation of software that will run on them. This evaluation ...

Keywords: runtime support, servers, simultaneous multithreading

19 Sapphire: copying GC without stopping the world

Richard L. Hudson, J. Eliot B. Moss
June 2001 **Proceedings of the 2001 joint ACM-ISCOPE conference on Java Grande**

Full text available: pdf(899.45 KB) Additional Information: full citation, abstract, references, citings, index terms

Many concurrent garbage collection (GC) algorithms have been devised, but few have been implemented and evaluated, particularly for the Java programming language. Sapphire is an algorithm we have devised for concurrent copying GC. Sapphire stresses minimizing the amount of time any given application thread may need to block to support the collector. In particular, Sapphire is intended to work well in the presence of a large number of application threads, on small- to medium-scale shared memor ...

20 The Amber system: parallel programming on a network of multiprocessors

J. Chase, F. Amador, E. Lazowska, H. Levy, R. Littlefield
November 1989 **ACM SIGOPS Operating Systems Review , Proceedings of the twelfth ACM symposium on Operating systems principles**, Volume 23 Issue 5

Full text available: pdf(1.53 MB) Additional Information: full citation, abstract, references, citings, index terms

This paper describes a programming system called Amber that permits a single application program to use a homogeneous network of computers in a uniform way, making the network appear to the application as an integrated multiprocessor. Amber is specifically designed for high performance in the case where each node in the network is a shared-memory multiprocessor. Amber shows that support for loosely-coupled multiprocessing can be efficiently realized using an obje ...

Results 1 - 20 of 41 Result page: **1** 2 3 next

Terms used **local** and **global** and **thread stack** and **heap** Found **41** of **148,786**

Sort results by relevance

Display results expanded form

Save results to a Binder

Search Tips

Open results in a new window

Try an Advanced Search

Try this search in The ACM Guide

Results 21 - 40 of 41 Result page: previous 1 **2** 3

Relevance scale

21 Type-based hot swapping of running modules (extended abstract)

Dominic Duggan

October 2001 **ACM SIGPLAN Notices , Proceedings of the sixth ACM SIGPLAN international conference on Functional programming**, Volume 36 Issue 10

Full text available: pdf(150.34 KB) Additional Information: full citation, abstract, references, citings, index terms

While dynamic linking has become an integral part of the run-time execution of modem programming languages, there is increasing recognition of the need for support for hot swapping of running modules, particularly in long-lived server applications. The interesting challenge for such a facility is to allow the new module to change the types exported by the original module, while preserving type safety. This paper describes a type-based approach to hot swapping running modules. The approach is bas ...

Keywords: dynamic typing, hot swapping, module interconnection languages, shared libraries

22 Implementation of Argus

B. Liskov, D. Curtis, P. Johnson, R. Scheifer

November 1987 **ACM SIGOPS Operating Systems Review , Proceedings of the eleventh ACM Symposium on Operating systems principles**, Volume 21 Issue 5

Full text available: pdf(1.34 MB) Additional Information: full citation, abstract, references, citings, index terms

Argus is a programming language and system developed to support the construction and execution of distributed programs. This paper describes the implementation of Argus, with particular emphasis on the way we implement atomic actions, because this is where Argus differs most from other implemented systems. The paper also discusses the performance of Argus. The cost of actions is quite reasonable, indicating that action systems like Argus are practical.

23 A parallel, incremental and concurrent GC for servers

Yoav Ossia, Ori Ben-Yitzhak, Irit Goft, Elliot K. Kolodner, Victor Leikehman, Avi Owshanko

May 2002 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2002 Conference on Programming language design and implementation**, Volume 37 Issue 5

Full text available: pdf(231.80 KB) Additional Information: full citation, abstract, references, citings, index terms

Multithreaded applications with multi-gigabyte heaps running on modern servers provide new challenges for garbage collection (GC). The challenges for "server-oriented" GC include: ensuring short pause times on a multi-gigabyte heap, while minimizing throughput penalty, good scaling on multiprocessor hardware, and keeping the number of expensive multi-cycle fence instructions required by weak ordering to a minimum. We designed and implemented a fully parallel, incremental, mostly concurrent colle ...

Keywords: JVM, Java, concurrent garbage collection, garbage collection, incremental garbage collection, weak ordering

24 Open runtime platform: flexibility with performance using interfaces

Michal Cierniak, Brian T. Lewis, James M. Stichnoth

November 2002 **Proceedings of the 2002 joint ACM-ISCOPE conference on Java Grande**

Full text available: pdf(300.99 KB) Additional Information: full citation, abstract, references, citings, index terms

According to conventional wisdom, interfaces provide flexibility at the cost of performance. Most high-performance Java virtual machines today tightly integrate their core virtual machines with their just-in-time compilers and garbage collectors to get the best performance. The Open Runtime Platform (ORP) is unusual in that it reconciles high performance with the extensive use of well-defined interfaces between its components. ORP was developed to support experiments in dynamic compilation, garb ...

Keywords: JVM, Java, dynamic compilation, garbage collection, interface design, interfaces, just-in-time compilation, modular components, virtual machine

25 Supporting thousands of threads using a hybrid stack sharing scheme

Kam-Fai Wong, Benoît Dageville

April 1994 **Proceedings of the 1994 ACM symposium on Applied computing**

Full text available: pdf(735.97 KB) Additional Information: full citation, references, index terms

26 Space efficient conservative garbage collection

Hans-Juergen Boehm

June 1993 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1993 conference on Programming language design and implementation**, Volume 28 Issue 6

Full text available: pdf(1.03 MB) Additional Information: full citation, abstract, references, citings, index terms

We call a garbage collector conservative if it has only partial information about the location of pointers, and is thus forced to treat arbitrary bit patterns as though they might be pointers, in at least some cases. We show that some very inexpensive, but previously unused techniques can have dramatic impact on the effectiveness of conservative garbage collectors in reclaiming memory. Our most significant observation is that static data that appears to point to the heap should not result i ...

27 Java without the coffee breaks: a nonintrusive multiprocessor garbage collector

David F. Bacon, Clement R. Attanasio, Han B. Lee, V. T. Rajan, Stephen Smith

May 2001 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2001 conference on Programming language design and implementation**, Volume 36 Issue 5

Full text available: pdf(1.69 MB) Additional Information: full citation, abstract, references, citings, index terms

The deployment of Java as a concurrent programming language has created a critical need

for high-performance, concurrent, and incremental multiprocessor garbage collection. We present the *Recycler*, a fully concurrent pure reference counting garbage collector that we have implemented in the Jalapeño Java virtual machine running on shared memory multiprocessors.

While a variety of multiprocessor collectors have been proposed and some have been implemented, experimental dat ...

28 Compiling nested data-parallel programs for shared-memory multiprocessors
Siddhartha Chatterjee
July 1993 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 15 Issue 3
Full text available: pdf(4.17 MB) Additional Information: full citation, references, citings, index terms, review

Keywords: compilers, data parallelism, shared-memory multiprocessors

29 The Flux OSKit: a substrate for kernel and language research
Bryan Ford, Godmar Back, Greg Benson, Jay Lepreau, Albert Lin, Olin Shivers
October 1997 **ACM SIGOPS Operating Systems Review , Proceedings of the sixteenth ACM symposium on Operating systems principles**, Volume 31 Issue 5
Full text available: pdf(2.47 MB) Additional Information: full citation, references, citings, index terms

30 OpenMP on networks of workstations
Honghui Lu, Y. Charlie Hu, Willy Zwaenepoel
November 1998 **Proceedings of the 1998 ACM/IEEE conference on Supercomputing (CDROM)**
Full text available: pdf(202.91 KB) Additional Information: full citation, abstract, references, citings

We describe an implementation of a sizable subset of OpenMP on networks of workstations (NOWs). By extending the availability of OpenMP to NOWs, we overcome one of its primary drawbacks compared to MPI, namely lack of portability to environments other than hardware shared memory machines. In order to support OpenMP execution on NOWs, our compiler targets a software distributed shared memory system (DSM) which provides multi-threaded execution and memory consistency.This paper presents two contri ...

31 An efficient meta-lock for implementing ubiquitous synchronization
Ole Agesen, David Detlefs, Alex Garthwaite, Ross Knippel, Y. S. Ramakrishna, Derek White
October 1999 **ACM SIGPLAN Notices , Proceedings of the 14th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications**, Volume 34 Issue 10
Full text available: pdf(2.00 MB) Additional Information: full citation, abstract, references, citings, index terms

Programs written in concurrent object-oriented languages, especially ones that employ thread-safe reusable class libraries, can execute synchronization operations (lock, notify, etc.) at an amazing rate. Unless implemented with utmost care, synchronization can become a performance bottleneck. Furthermore, in languages where every object may have its own monitor, per-object space overhead must be minimized. To address these concerns, we have developed a meta-lock to mediate access to synchro ...

Keywords: concurrent threads, object-oriented language implementation, synchronization

32 A high performance Erlang system
Erik Johansson, Mikael Pettersson, Konstantinos Sagonas
September 2000 **Proceedings of the 2nd ACM SIGPLAN international conference on Principles and practice of declarative programming**
Full text available: pdf(320.62 KB) Additional Information: full citation, references, citings, index terms

33 Efficient process interaction with threads in parallel discrete event simulation
Reuben Passqini, Vernon Rego
December 1998 **Proceedings of the 30th conference on Winter simulation**
Full text available: pdf(80.65 KB) Additional Information: full citation, references, index terms

34 Scheduling threads for low space requiremeni and good locality
Girija J. Narlikar
June 1999 **Proceedings of the eleventh annual ACM symposium on Parallel algorithms and architectures**
Full text available: pdf(1.69 MB) Additional Information: full citation, references, citings, index terms

35 Adaptive two-level thread management for fast MPI execution on shared memory machines
Kai Shen, Hong Tang, Tao Yang
January 1999 **Proceedings of the 1999 ACM/IEEE conference on Supercomputing (CDROM)**
Full text available: pdf(152.63 KB) Additional Information: full citation, references, citings, index terms

36 Implementing jalapeño in Java
Bowen Alpern, C. R. Attanasio, Anthony Cocchi, Derek Lieber, Stephen Smith, Ton Ngo, John J. Barton, Susan Flynn Hummel, Janice C. Sheperd, Mark Mergen
October 1999 **ACM SIGPLAN Notices , Proceedings of the 14th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications**, Volume 34 Issue 10
Full text available: pdf(1.57 MB) Additional Information: full citation, abstract, references, citings, index terms

Jalapeño is a virtual machine for Java™ servers written in Java.A running Java program involves four layers of functionality: the user code, the virtual-machine, the operating system, and the hardware. By drawing the Java / non-Java boundary below the virtual machine rather than above it, Jalapeño reduces the boundary-crossing overhead and opens up more opportunities for optimization.To get Jalapeño started, a boot image of a ...

37 Pthreads for dynamic and irregular parallelism
Girija J. Narlikar, Guy E. Blelloch
November 1998 **Proceedings of the 1998 ACM/IEEE conference on Supercomputing (CDROM)**
Full text available: html(82.60 KB) Additional Information: full citation, abstract, references, citings

High performance applications on shared memory machines have typically been written in a coarse grained style, with one heavyweight thread per processor. In comparison,

programming with a large number of lightweight, parallel threads has several advantages, including simpler coding for programs with irregular and dynamic parallelism, and better adaptability to a changing number of processors. The programmer can express a new thread to execute each individual parallel task; the implementation dyn ...

Keywords: Pthreads, dynamic scheduling, irregular parallelism, lightweight threads, multithreading, space efficiency

38 Clarity MCode: a retargetable intermediate representation for compilation 
Brian T. Lewis, L. Peter Deutsch, Theodore C. Goldstein
March 1995 **ACM SIGPLAN Notices , Papers from the 1995 ACM SIGPLAN workshop on Intermediate representations**, Volume 30 Issue 3
Full text available: pdf(948.64 KB) Additional Information: full citation, citings, index terms

39 Access rights analysis for Java 
Larry Koved, Marco Pistoia, Aaron Kershenbaum
November 2002 **ACM SIGPLAN Notices , Proceedings of the 17th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications**, Volume 37 Issue 11
Full text available: pdf(360.93 KB) Additional Information: full citation, abstract, references, citings, index terms

Java 2 has a security architecture that protects systems from unauthorized access by mobile or statically configured code. The problem is in manually determining the set of security access rights required to execute a library or application. The commonly used strategy is to execute the code, note authorization failures, allocate additional access rights, and test again. This process iterates until the code successfully runs for the test cases in hand. Test cases usually do not cover all paths th ...

Keywords: Java security, access rights, call graph, data flow analysis, invocation graph, security

40 Systems, platforms, and applications: MANTIS: system support for multimodAl NeTworks of in-situ sensors 
H. Abrach, S. Bhatti, J. Carlson, H. Dai, J. Rose, A. Sheth, B. Shucker, J. Deng, R. Han
September 2003 **Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications**
Full text available: pdf(424.53 KB) Additional Information: full citation, abstract, references, citings, index terms

The *MANTIS M*ultimod*A*l system for *NeT*works of In-situ wireless Sensors provides a new multithreaded embedded operating system integrated with a general-purpose single-board hardware platform to enable flexible and rapid prototyping of wireless sensor networks. The key design goals of MANTIS are ease of use, i.e. a small learning curve that encourages novice programmers to rapidly prototype novel sensor networking applications in software and hardware, as well as flexibility, ...

Keywords: GPS, dynamic reprogramming, lightweight, multimodal prototyping, operating systems, wireless sensor networks

Terms used **local** and **global** and **thread stack** and **heap** Found **41** of **148,786**

Sort results by: relevance
Display results: expanded form
Save results to a Binder
Search Tips
Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 41 - 41 of 41 Result page: previous 1 2 **3**

Relevance scale

41 Optimistic active messages: a mechanism for scheduling communication with computation

Deborah A. Wallach, Wilson C. Hsieh, Kirk L. Johnson, M. Frans Kaashoek, William E. Weihl
August 1995 **ACM SIGPLAN Notices , Proceedings of the fifth ACM SIGPLAN symposium on Principles and practice of parallel programming**, Volume 30 Issue 8

Full text available: pdf(1.14 MB) Additional Information: full citation, abstract, references, citings, index terms

Low-overhead message passing is critical to the performance of many applications. Active Messages reduce the software overhead for message handling: messages are run as handlers instead of as threads, which avoids the overhead of thread management and the unnecessary data copying of other communication models. Scheduling the execution of Active Messages is typically done by disabling and enabling interrupts, or by polling the network. This primitive scheduling control, combined with the fac ...

Results 41 - 41 of 41 Result page: previous 1 2 **3**

Google

Web Images Groups[New!] News Froogle more »

"thread stack" local global "thread heap" sta Search Advanced Search Preferences

Web

Results **1** - **5** of **5** for "**thread stack**" **local global** "**thread heap**" **status variable**. (**0.28** seconds)

Tip: Try removing quotes from your search to get more results.

[PDF] Developing Platform Consistent Multithreaded Applications: Memory ...
File Format: PDF/Adobe Acrobat - View as HTML
... TLS but the management of this **global** storage is ... images and access patterns to **local** variables on ... Adjusting the initial **thread stack** address of each thread is ...
cache-www.intel.com/cd/00/00/ 05/15/51533_chapter_5_memory_management02.pdf - Similar pages

[PS] Computer Systems Laboratory Cornell University, Ithaca, NY 14853
File Format: Adobe PostScript - View as Text
... thread-specific instances of all **global** variables utilizing ... the address of a thread-**local variable** is passed ... a function call, it is pushed to the **thread stack**. ...
www.csl.cornell.edu/TR/CSL-TR-2001-1016.ps - Similar pages

[PS] Adding Threads to Standard ML
File Format: Adobe PostScript - View as Text
... of thread identifier and thread-**local** variables. ... a way that by masking a **thread's stack** pointer appropriately ... A thread's state vector and "**global**" variables are ...
www.eecs.harvard.edu/~greg/papers/jgmorris-mlthreads.ps - Similar pages

[PS] Region-Based Memory Management for
File Format: Adobe PostScript - View as Text
... Thread **local** state implementations of the specification may ... the start of the thread (**thread stack** entry point ... The program updates the **status variable** using non ...
flexc.lcs.mit.edu/Harpoon/papers/wes-thesis.ps - Similar pages

[PS] Resource Control of Untrusted Code in an
File Format: Adobe PostScript - View as Text
... for Active Network Ser- vices" [Menage99] in the First International Working Conference on ... the router or **local** network, or to perform some control operation. ...
www.paulmenage.org/papers/thesis.ps - Similar pages

Google Web Images Groups[New] News Froogle more »
"thread stack" local global "thread heap" st Search Advanced Search Preferences

Web

Results 1 - 7 of 7 for "thread stack" local global "thread heap" status. (0.30 seconds)

Tip: Try removing quotes from your search to get more results.

[PDF] Developing Platform Consistent Multithreaded Applications: Memory ...
File Format: PDF/Adobe Acrobat - View as HTML
... TLS but the management of this **global** storage is ... images and access patterns to **local** variables on ... Adjusting the initial **thread stack** address of each thread is ...
cache-www.intel.com/cd/00/00/ 05/15/51533_chapter_5_memory_management02.pdf - Similar pages

[PDF] A Foundation 345
File Format: PDF/Adobe Acrobat
... jects (ie, objects resident in a **global** address space) into ... **heap**; an attempt to write onto this page triggers ... s because heaps and stacks are **local** to threads ...
portal.acm.org/ft_gateway.cfm?id=141573&type=pdf - Similar pages

[PS] Computer Systems Laboratory Cornell University, Ithaca, NY 14853
File Format: Adobe PostScript - View as Text
... thread-specific instances of all **global** variables utilizing ... When the address of a thread-**local** variable is ... a function call, it is pushed to the **thread stack**. ...
www.csl.cornell.edu/TR/CSL-TR-2001-1016.ps - Similar pages

[PS] Adding Threads to Standard ML
File Format: Adobe PostScript - View as Text
... of thread identifier and thread-**local** variables. ... a way that by masking a **thread's stack** pointer appropriately ... A thread's state vector and "**global**" variables are ...
www.eecs.harvard.edu/~greg/papers/jgmorris-mlthreads.ps - Similar pages

[PS] Region-Based Memory Management for
File Format: Adobe PostScript - View as Text
... by looking at the **thread's stack** of scoped memories ... primitives, optimistic synchro- nization, **status** variables, and ... Thread **local** state implementations of the ...
flexc.lcs.mit.edu/Harpoon/papers/wes-thesis.ps - Similar pages

[PS] Resource Control of Untrusted Code in an
File Format: Adobe PostScript - View as Text
... for Active Network Ser- vices" [Menage99] in the First International Working Conference on ... the router or **local** network, or to perform some control operation. ...
www.paulmenage.org/papers/thesis.ps - Similar pages

kernel.h File Reference
... size should include space for the **thread stack** and **local** ... that was previously allocated on the **global** heap by a ... previously allocated on the thread **local** heap by ...
www.cloudcaptech.com/Datasheets/uBurst/kernel_8h.html - 94k - Supplemental Result - Cached - Similar pages

Free! Google Desktop Search: Search your own computer.